РЕЛЯЦІЙНІ БАЗИ ДАНИХ

SQL

База даних (БД) – це впорядкований набір логічно взаємопов'язаних даних, призначених для задоволення інформаційних потреб певної предметної області.

Доступ до даних БД здійснюється різними програмами через посередництво систем керування базами даних (СКБД).

Система керування базами даних (СКБД) — сукупність програмних засобів, що забезпечують керування створенням та використанням баз даних.

СКБД, розміщена на одному комп'ютері називається *локальною* СКБД. СКБД, частини якої можуть розміщатися на декількох комп'ютерах називається *розподіленою*.

Дані в БД можна представляти по-різному, в залежності від методів опису типів та логічних структур даних в БД, методів маніпуляції даними, методів опису та підтримки цілісності БД (того, що дані не будуть змінені при виконанні таких операцій над ними, як то представлення, збереження, передача, представлення).

Модель даних – це абстрактне, самодостатнє представлення об‘єктів, операторів та інших елементів, які в сукупності складають абстрактну машину доступу до даних, з якими взаємодіє користувач. Ці об’єкти дозволяють моделювати структуру даних, а оператори – поведінку даних.

Моделі даних (МД):

Виділяють наступні моделі даних (МД):

Ієрархічна МД – представляє БД у вигляді ієрархічної (деревовидної) структури. Об’єкти знаходяться у відношенні «нащадок» – «предок». Перші СКБД використовували ієрархічну модель представлення даних.

Мережева МД – представляє БД у вигляді вузлів (елементів), зв’язків між ними на різних рівнях. Мережева МД є розширенням ієрархічної шляхом введення вказівників, що з’єднують споріднену інформацію в обох напрямках.

Реляційна МД – представляє БД у вигляді таблиць, що надає можливість використання формального апарата алгебри відношень та реляційного числення для обробки даних. Реляційна модель передбачає, що кожен елемент таблиці є одним елементом даних, кожен стовпчик має унікальне ім’я та містити дані одного типу, відсутність однакових рядків в таблиці, довільний порядок рядків та стовпчиків.

Об’єктно-орієнтована МД – розглядає дані як абстрактні наділені певними властивостями об’єкти, що використовують методи взаємодії з іншими об’єктами предметної області.

Об’єктно-реляційна МД – реляційна модель МД, яка підтримує деякі технології об’єктно-орієнтованих моделей.

Відповідно до МД СКБД поділяють на *ієрархічні*, *мережеві*, *реляційні*, *об'єктно-орієнтовані* та *об'єктно-реляційні* СКБД. На сьогоднішній день найбільшого розповсюдження набули об'єктно-реляційні СКБД. Прикладами таких СКБД є Oracle Database, Informix, DB2, PostgreSQL. Спільною для усіх перерахованих СКБД є реляційна складова.

Реляційні БД

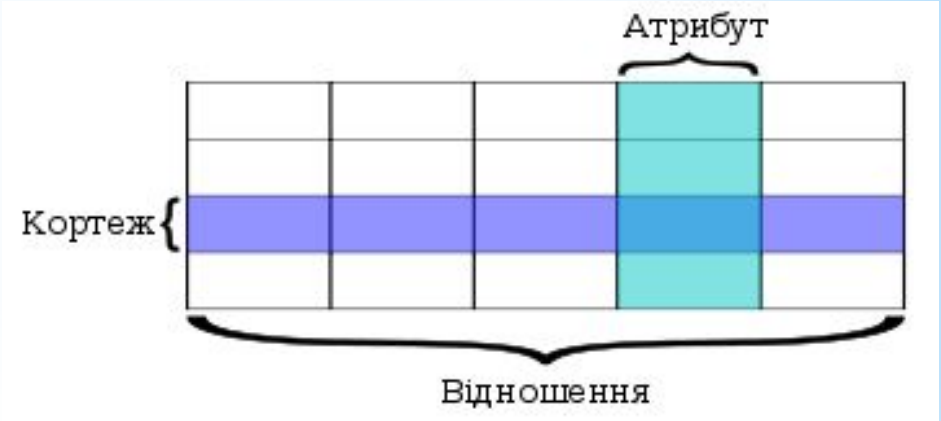
Реляційна моделі даних – це модель в якій дані організуються у вигляді набора таблиць.

Використання реляційних БД було запропоноване Едгаром Коддом в 1970-і роки.

Реляція (таблиця) позначається, як $R(A_1,...A_n)$, де R - ім'я реляції. $A_1,...A_n$ - імена **атрибутів (полів)**. Вони мають бути унікальними в межах однієї реляції. Порядок атрибутів несуттєвий, через унікальність імен.

Приклад: КНИГА(ІД, НАЗВА, ОПИС).

Інші терміни



Кортеж - рядок, запис.

Атрибут - стовпець, поле.

Степінь - кількість стовпців.

Кардинальність - кількість рядків.

Первинний ключ - ідентифікатор запису.

Домен - область допустимих значень. Тип даних.

Над цією структурою даних існує реляційна алгебра.

Атрибути A і B називають порівнюваними, якщо до будь-якої пари значень з доменів цих атрибутів можна застосувати предикат порівняння.

Набори атрибутів L та M називають порівнюваними, якщо можна встановити таку бієкцію між атрибутами, що кожна пара буде порівнювана.

Для зручності вважатимемо, що зіставлені атрибути порівнюваних відношень повинні мати однакові імена.

Над реляційною структурою даних будується реляційна алгебра. Її елементами є **реляції** (**таблиці**).

Сигнатура реляційної алгебри містить наступні операції: об'єднання, перетин, різниця, вибірка (обмеження), проекція, декартів добуток, з'єднання, ділення, переіменування. Результатами виконання цих операцій є реляції (замкненість реляційної алгебри).

$$R = f(f_1(R_{11}, R_{12}, \ldots), f_2(R_{21}, R_{22}, \ldots), \ldots)$$

Декартів добуток. Отношение $(A_1, A_2, \ldots, A_m, B_1, B_2, \ldots, B_m)$, яке є зціпленням відношень $A(A_1, A_2, \ldots, A_m)$ и $B(B_1, B_2, \ldots, B_m)$:

$(a_1, a_2, \ldots, a_m, b_1, b_2, \ldots, b_m)$таких, что

$(a_1, a_2, \ldots, a_m) \in A, (b_1, b_2, \ldots, b_m) \in B$.

Вибірка (обмеження). **Узагальнена вибірка** це унарний оператор, що записується як $\sigma_{\varphi}(R)$, де φ є формулою числення висловлювань, що складається із атомів, дозволених у звичайній вибірці та логічних операторів (кон'юнкції), (диз'юнкції) та (заперечення). Така вибірка вибирає всі кортежі для яких значення φ істина.

Проекція. Проекція — це унарна операція $\pi_{a1,\dots an}(R)$, де $a_1, \dots \ a_n$ є множиною назв атрибутів. Результат проекції визначається, як множина, що отримується із всіх кортежів із R, що обмежуються $\{a_1, \dots \ a_n\}$.

Переіменування. **Перейменування** є унарним оператором $p_{a/b}(R)$. Результат застосування оператора ідентичний за винятком того, що поле в усіх кортежах перейменовується на поле.

Об’єднання. Відношення з тим же заголовком, що і у сумісних за типом відношень A та B, та тілом, що складається з кортежів, що належать A чи B, чи обом відношенням: $A \cup B$.

Перетин. Відношення з тим же заголовком, що і у сумісних за типом відношень A та B, та тілом, що складається з кортежів, що належать A та B: $A \cap B$.

Різниця. Відношення з тим же заголовком, що і у сумісних за типом відношень A та B, та тілом, що складається з кортежів, що належать A та B не належать B: A/B.

З’єднання. Результат послідовного застосування операцій декартового добутку та вибірки. Якщо в відношеннях є атрибути з одинаковими наіменуваннями, то перед виконанням з’єднання їх необхідно переіменувати.

Ділення. Відношення з заголовком $(X_1, X_2, \ldots, X_n)$ та тілом, що містить множину кортежів $(x_1, x_2, \ldots, x_n)$, таких, що для усіх кортежів $(y_1, y_2, \ldots, y_m) \in B$ у відношенні $A(X_1, X_2, \ldots, X_n, Y_1, Y_2, \ldots, Y_m)$ знайдеться кортеж $(x_1, x_2, \ldots, x_n, y_1, y_2, \ldots, y_m)$.

У кожної таблиці є унікальне ім’я.

В кожній таблиці є один, або більше стовпчиків, кожен з яких має визначений тип та своє унікальне (в межах таблиці) ім’я.

В кожній таблиці є нуль, або більше рядків, кожен з яких містить одне значення даних в кожному стовпчику, при чому це значення має тип свого рядка.

Рядки в таблиці не впорядковані.

Відношення між таблицями реалізуються за первинних та зовнішніх ключів.

Первинний ключ

В правильно побудованій реляційній БД в кожній таблиці є один, або декілька стовпчиків, значення яких в усіх рядках унікальні. Цей стовпчик (чи стовпчики) називається *первинним ключем таблиці*. На практиці, в якості первинного ключа, як правило, слід обирати ідентифікатори, такі як номер рейсу, номер потягу, тощо.

Зовнішні ключі

Стовпчик однієї таблиці, значення в якому співпадає зі значенням стовпчика, що є первинним ключем іншої таблиці, називається *зовнішнім ключем*.

Зовнішній ключ, як і первинний, може являти собою комбінацію декількох стовпчиків. Зовнішні ключі слугують для зв'язування кількох таблиць. Якщо таблиця пов'язана з кількома іншими таблицями, то вона може мати декілька зовнішніх ключів.

Пара «первинний ключ – зовнішній ключ» задає відношення предок/нащадок між таблицями, що їх містять.

SQL. Стандарти SQL

SQL. Мовою реляційних, а відповідно і об'єктно-реляційних БД є мова SQL. Ця мова базується на численні кортежів.

Стандарти SQL. Одним з найважливіших кроків з визнання SQL на ринку стала його стандартизація. Перший стандарт SQL-86, прийнятий інститутом ANSI та підтриманий ISO з'явився в 1986 році. Після цього світ побачили наступні стандарти: SQL-89, SQL-92, SQL:1999, SQL:2003, SQL:2006, SQL:2008.

Рівні відповідності

Рівні відповідності. Починаючи з SQL:1999 стандарт має модульну структуру. Основна частина стандарту внесена в розділ «SQL/Foundation», всі інші його частини винесені в окремі модулі. Таким чином, усі СКБД повинні підтримувати основну частину (Core), а інші частини стандартів підтримуються на розгляд виробника СКБД. Далі, розглянемо лише можливості SQL/Foundation.

Короткий огляд мови SQL

Для часто виконуваних запитів, або як для допомоги в написанні складних запитів зручно зберігати його як збережений запит, яка складається з результатів цього запиту. Такий запит називається представленням (**View**).

Представлення (**View**) – це збережений запит, доступний як віртуальна динамічна таблиця, що складається з результатів запиту. Зміна даних в таблицях БД змінює їх у відповідних представленнях.

Транзакція – це декілька послідовних операторів SQL, які розглядаються як єдине ціле. В транзакції кожен оператор розв'язує частину загальної задачі, але для розв' язання усієї задачі, потрібно, щоб усі ці оператори були виконані. У випадку виникнення проблеми виникає відкат транзакції, тобто не виконується вся послідовність операторів.

Часто, при настанні певних подій в БД (наприклад, додавання чи оновлення видалення) потрібно виконувати певні процедури. Наприклад, при видаленні певного товару в магазині, можливо потрібно видалити всі не виконані замовлення на цей товар. Для таких дій використовуються тригери.

Тригер – це збережена процедура, використання якої обумовлено настанням визначеної події у БД, як: додавання, вилучення чи зміна даних. Тригери застосовуються для забезпечення цілісності даних. Тригер запускається сервером автоматично при настанні події з якою він пов'язаний. Всі здійснені ним модифікації даних розглядаються як виконані в транзакції.

Основні типи даних, визначені в стандаритах SQL

CHAR(len)	- рядок символів фіксованої довжини розміром len символів;
VARCHAR(len)	- рядок символів змінної довжини з максимальним розміром до len символів;
INT[(len)]	- ціле число довжиною 4 байта, що представляється при виведенні максимально len цифрами;
SMALLINT[(len)]	- ціле число довжиною 2 байта, що представляється при виведенні максимально len цифрами;
FLOAT[(len, dec)]	- дійсне число, де len – це максимальна кількість символів, dec – кількість цифр після десяткової точки;

DATE	**- календарна дата;**
TIME	- астрономічний час
NUMERIC (len, dec) (NUMBER(len, dec))	- дійсне число, де len – це максимальна кількість символів, dec – кількість цифр після десяткової точки;
NUMERIC (len) (NUMBER(len))	- ціле число, яке має максимальну довжину len.

Рядкові та символьні константи повинні міститися в одинарних чи подвійних лапках.

УВАГА!!! **Назви типові даних в різних СКБД не завжди співпадають між собою та з рекомендаціями стандарту SQL.**

Вирази в SQL

Вирази в SQL використовуються для виконання операцій над значеннями, прочитаними з БД чи тими, що використовуються для пошуку в БД. Вирази можуть включати в себе дужки, арифметичні операції +, -, *, / та деякі функції:

UPPER(рядок)	**-перетворення рядка до верхнього регістра**
LOWER(рядок)	- перетворення рядка до нижнього регістра
DAY(дата)	- повертає число дати
MONTH(дата)	- повертає місяць дати
YEAR(дата)	- повертає рік дати
GETDATE()	- повертає поточну системну дати
LEN(рядок)	- повертає довжину рядка в символах
ROUND(число, точність)	– виконує округлення числа з потрібною точністю

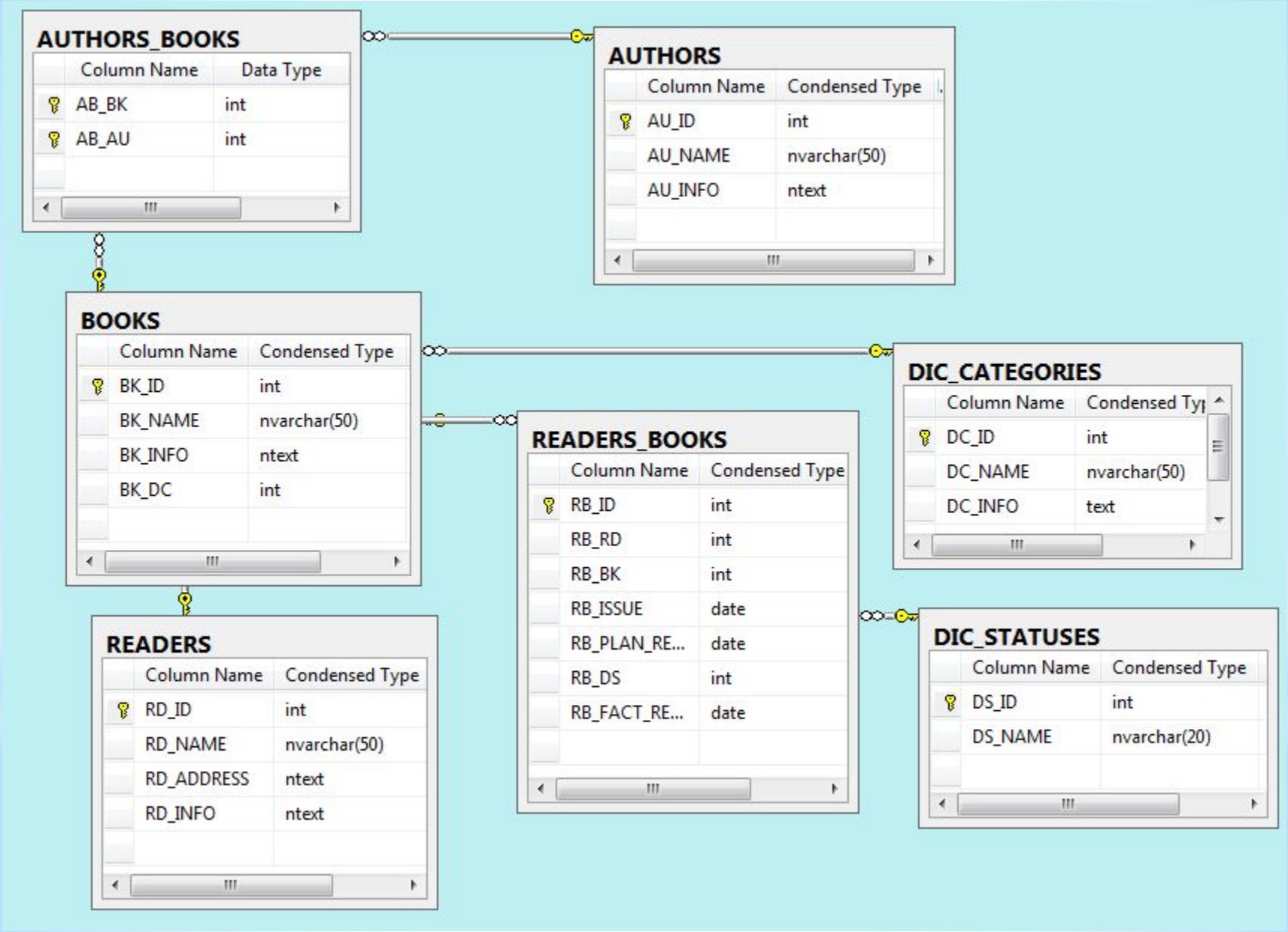
Крім цього, використовуються *агрегатні функції*, які дозволяють отримувати різні види статистичної інформації, використовуючи в якості аргументу деякий стовпчик повністю, а повертаючи одне значення.

COUNT([DISTINCT] [*][<вираз>])	**- підраховує кількість рядків в групах. Якщо вказати * чи довільну константу, крім NULL, то функція буде рахувати всі рядки в таблиці результатів. Якщо задати вираз, то рядки, для яких вираз має значення не NULL. Задання модифікатора DISTINCT надає команду рахувати рядки в групах після видалення дублюючих рядків.**
SUM(<вираз>| DISTINCT <ім'я_стовпчика>)	- обчислює суму всіх значень, що містяться в стовпчику. Якщо задати вираз, то обчислюється сума значень стовпчика, для яких вираз має значення не NULL. Задання модифікатора DISTINCT надає команду рахувати суму значень після видалення дублюючих рядків.

AVG(<вираз>| DISTINCT <ім'я_стовпчика>)	- обчислює середнє всіх значень, що містяться в стовпчику. Якщо задати вираз, то обчислюється середнє значень стовпчика, для яких вираз має значення не NULL. Задання модифікатора DISTINCT надає команду рахувати середнє після видалення дублюючих рядків.
MAX(<вираз>)	- знаходить найбільше серед усіх значень, що містяться в стовпчику.
MIN(<вираз>)	- знаходить найменше серед усіх значень, що містяться в стовпчику.

Для складних виразів використовуються дужки.

Нехай маємо базу даних бібліотеки:



Деякі оператори SQL

Обробка даних	
SELECT	- зчитування даних з БД;
INSERT	- додавання нових рядків в БД;
DELETE	- видалення рядків з БД;
UPDATE	- обновлення даних в БД;
Визначення даних	
CREATE TABLE	- створення в БД нової таблиці;
DROP TABLE	- видалення таблиці з БД;
ALTER TABLE	- зміна структури існуючої таблиці;
CREATE VIEW	- додавання в БД нового представлення;
DROP VIEW	- видалення представлення з БД;
CREATE INDEX	- створення індексу для стовпчика;
DROP INDEX	- видалення індексу стовпчика;

Запит на створення таблиці CREATE TABLE

<create-table> ::= CREATE TABLE <ім’я-таблиці>
(<список-визначень-стовпчиків>)
[<визначення-первинних-ключів>]
[<визначення-зовнішніх-ключів>]
[<умова-унікальності-даних>]
[<умова-перевірки>]

Розглянемо більш детально секції запиту <create-table>

<список-визначень-стовпчиків> складається із

визначень стовпчиків через кому.

<визначення-стовпчика> ::= <ім’я-стовпчика> <тип-даних> [DEFAULT значення]

[DEFAULT значення] - означає значення по замовчуванню, а [NOT NULL] - означає, що відповідне значення не може бути не порожнім.

<визначення-первинних-ключів> ::=

PRIMARY KEY(<список-імен-стовпчиків-через-кому>)

```
<визначення-зовнішніх-ключів> ::=
        [<ім’я-відношення>]
        (<список-імен-стовпчиків-через-кому>)
        REFERENCE <ім’я-таблиці>
        [MATCH
            FULL | PARTIAL
        ]
        [ON DELETE
            CASCADE | SET NULL | SET DEFAULT | NO ACTION
        ]
        [ON UPDATE
            ` CASCADE | SET NULL | SET DEFAULT | NO ACTION
        ]
```

Тут, **REFERENCE <ім'я-таблиці>** задає таблицю-предок з якою відбувається зв'язування.

[ON DELETE
CASCADE | SET NULL | SET DEFAULT | NO ACTION
]
та
[ON UPDATE
CASCADE | SET NULL | SET DEFAULT | NO ACTION
] – задають правила поведінки при видаленні чи оновленні даних,

<умова-унікальності-даних> ::=
UNIQUE(<список-імен-стовпчиків-через-кому>)

<умова-перевірки> ::= CHECK(<умова-пошуку>)

Приклад 1 (MS SQL Server)

```
CREATE TABLE [AUTHORS](
    [AU_ID] [int] IDENTITY(1,1) NOT NULL,
    [AU_NAME] [nvarchar](50) NOT NULL,
    [AU_INFO] [ntext] NULL,
 CONSTRAINT [PK_AUTHORS_1] PRIMARY KEY CLUSTERED
(
    [AU_ID] ASC
))

CREATE TABLE [DIC_CATEGORIES](
    [DC_ID] [int] IDENTITY(1,1) NOT NULL,
    [DC_NAME] [nvarchar](50) NOT NULL,
    [DC_INFO] [text] NULL,
 CONSTRAINT [PK_DIC_CATEGORIES] PRIMARY KEY CLUSTERED
(
    [DC_ID] ASC
))
```

```
CREATE TABLE [DIC_STATUSES](
    [DS_ID] [int] NOT NULL,
    [DS_NAME] [nvarchar](20) NOT NULL,
 CONSTRAINT [PK_DIC_STATUSES] PRIMARY KEY CLUSTERED
( [DS_ID] ASC))

 CREATE TABLE [READERS](
    [RD_ID] [int] IDENTITY(1,1) NOT NULL,
    [RD_NAME] [nvarchar](50) NOT NULL,
    [RD_ADDRESS] [ntext] NULL,
    [RD_INFO] [ntext] NULL,
 CONSTRAINT [PK_READERS] PRIMARY KEY CLUSTERED
( [RD_ID] ASC ))
```

```
CREATE TABLE [BOOKS](
    [BK_ID] [int] IDENTITY(1,1) NOT NULL,
    [BK_NAME] [nvarchar](50) NOT NULL,
    [BK_INFO] [ntext] NULL,
    [BK_DC] [int] NOT NULL,
CONSTRAINT [PK_BOOKS] PRIMARY KEY CLUSTERED ([BK_ID] ASC),
CONSTRAINT [FK_BOOKS_DIC_CATEGORIES] FOREIGN KEY([BK_DC])
REFERENCES [DIC_CATEGORIES] ([DC_ID]))

CREATE TABLE [AUTHORS_BOOKS](
    [AB_BK] [int] NOT NULL,
    [AB_AU] [int] NOT NULL,
 CONSTRAINT [PK_AUTHORS_BOOKS] PRIMARY KEY CLUSTERED
([AB_BK] ASC, [AB_AU] ASC ),
CONSTRAINT [FK_AUTHORS_BOOKS_AUTHORS] FOREIGN KEY([AB_AU])
REFERENCES [AUTHORS] ([AU_ID]),
CONSTRAINT [FK_AUTHORS_BOOKS_BOOKS1] FOREIGN KEY([AB_BK])
REFERENCES [BOOKS] ([BK_ID]))
```

```
CREATE TABLE [READERS_BOOKS](
    [RB_ID] [int] IDENTITY(1,1) NOT NULL,
    [RB_RD] [int] NOT NULL,
    [RB_BK] [int] NOT NULL,
    [RB_ISSUE] [date] NOT NULL,
    [RB_PLAN_RETURN] [date] NOT NULL,
    [RB_DS] [int] NOT NULL,
    [RB_FACT_RETURN] [date] NULL,
 CONSTRAINT [PK_READERS_BOOKS] PRIMARY KEY CLUSTERED
([RB_ID] ASC),
 CONSTRAINT [FK_READERS_BOOKS_BOOKS1] FOREIGN KEY([RB_BK])
REFERENCES [BOOKS] ([BK_ID]),
 CONSTRAINT [FK_READERS_BOOKS_DIC_STATUSES] FOREIGN
KEY([RB_DS]) REFERENCES [DIC_STATUSES] ([DS_ID]),
 CONSTRAINT [FK_READERS_BOOKS_READERS] FOREIGN KEY([RB_RD])
REFERENCES [READERS] ([RD_ID]))
```

Створимо зайву таблицю:

```
CREATE TABLE [BOOKS1](
    [BK1_ID] [int] IDENTITY(1,1) NOT NULL,
    [BK1_INFO] [ntext] NULL,
CONSTRAINT [PK_BOOKS1] PRIMARY KEY CLUSTERED
([BK1_ID] ASC))
```

ЗАПИТ НА ЗМІНУ СТРУКТУРИ ІСНУЮЧОЇ ТАБЛИЦІ ALTER TABLE

<alter-table> ::= ALTER TABLE <ім'я-таблиці>

ADD <визначення-стовпчика>|

ALTER <ім'я-стовпчика> SET DEFAULT <значення>| DROP DEFAULT | DROP <ім'я-стовпчика> CASCADE|RESTRICT |

ADD [<визначення-первинних-ключів>]

[<визначення-зовнішніх-ключів>]

[<умова-унікальності-даних>]

[<умова-перевірки>]

| DROP CONSTRAINT <умова> CASCADE|RESTRICT

Секції аналогічні до секцій запиту на створення таблиці.

Змінимо структуру таблиці BOOKS, додавши до неї новий стовпчик:

Приклад 2.

```
ALTER TABLE [BOOKS1]
ADD  [BK1_NAME] [nvarchar](50) NOT NULL
```

ЗАПИТ НА ВИДАЛЕННЯ ТАБЛИЦІ DROP TABLE

<drop-table> ::= DROP TABLE <ім'я-таблиці> [CASCADE | RESTRICT]

Розглянемо більш детально секції запиту <drop-table>

Якщо задано параметр **CASCADE**, і в БД існують об'єкти, які містять посилання на видаляємо таблицю (містять відповідний зовнішній ключ), то видалення не відбудеться.

Для видалення таблиці **BOOKS1** з бази даних бібліотеки створимо наступний запит.

Приклад 3.

DROP TABLE [BOOKS1]

ЗАПИТ НА ДОДАВАННЯ НОВИХ РЯДКІВ INSERT

<insert> ::= INSERT INTO <ім'я-таблиці> [(<список-стовпчиків>)]
VALUES (<список-значень-для-стовпчиків>)

<список-значень-для-стовпчиків> ::=

<КОНСТАНТА> | <NULL> |

<константа>, <список-значень-для-стовпчиків>
|<NULL>, <список-значень-для-стовпчиків>

Виконаємо послідовно наступні запити на додавання нових записів.

```
INSERT INTO [AUTHORS]
([AU_NAME], [AU_INFO]) VALUES ('Зубенко В.В.', '')

INSERT INTO [AUTHORS]
([AU_NAME], [AU_INFO]) VALUES ('Омельчук Л.Л.', '')

INSERT INTO [AUTHORS]
([AU_NAME], [AU_INFO]) VALUES ('Нікітченко М.С.', '')

INSERT INTO [AUTHORS]
([AU_NAME], [AU_INFO]) VALUES ('Шкільняк С.С.', '')

INSERT INTO [AUTHORS]
([AU_NAME], [AU_INFO]) VALUES ('Лавріщева К.М.', '')
```

[AUTHORS]

AU_ID	AU_NAME	AU_INFO
1	Зубенко В.В.	
2	Омельчук Л.Л.	
3	Нікітченко М.С.	
4	Шкільняк С.С.	
5	Лавріщева К.М.	

```
INSERT INTO [DIC_CATEGORIES]
([DC_NAME])
VALUES ('Програмування')

INSERT INTO [DIC_CATEGORIES]
([DC_NAME])
VALUES ('Логіка')
```

[DIC_CATEGORIES]

	DC_ID	DC_NAME	DC_INFO
▶	1	Програмування	*NULL*
	2	Логіка	*NULL*
*	*NULL*	*NULL*	*NULL*

```
INSERT INTO [BOOKS]
([BK_NAME], [BK_INFO], [BK_DC])
VALUES ('Програмування', 'ІНФОРМАЦІЯ Програмування',1)

INSERT INTO [BOOKS]
([BK_NAME], [BK_INFO], [BK_DC])
VALUES ('Програмна інженерія', 'ІНФОРМАЦІЯ Програмна
інженерія', 1)

INSERT INTO [BOOKS]
([BK_NAME], [BK_INFO], [BK_DC])
VALUES ('Математична логіка та теорія алгоритмів',
'ІНФОРМАЦІЯ Математична логіка та теорія алгоритмів', 2)
```

[BOOKS]

BK_ID	BK_NAME	BK_INFO	BK_DC
1	Програмування	ІНФОРМАЦІЯ Програмування	1
2	Програмна інженерія	ІНФОРМАЦІЯ Програмна інженерія	1
3	Математична логіка та теорія алго...	ІНФОРМАЦІЯ Математична логіка та те...	2
NULL	*NULL*	*NULL*	*NULL*

```
INSERT INTO [AUTHORS_BOOKS] ([AB_AU], [AB_BK])
VALUES (1, 1)

INSERT INTO [AUTHORS_BOOKS] ([AB_AU], [AB_BK])
VALUES (2, 1)

INSERT INTO [AUTHORS_BOOKS] ([AB_AU], [AB_BK])
VALUES (3, 3)

INSERT INTO [AUTHORS_BOOKS] ([AB_AU], [AB_BK])
VALUES (4, 3)

INSERT INTO [AUTHORS_BOOKS] ([AB_AU], [AB_BK])
VALUES (5, 2)
```

[AUTHORS_BOOKS]

AB_BK	AB_AU
1	1
1	2
2	3
3	4
3	5

ЗАПИТ НА ПОШУК SELECT

<select> ::= SELECT [ALL | DISTINCT | DISTINCTROW]
<список-повертаємих-стовпчиків> | *
FROM <список-таблиць>
[WHERE <умова-пошуку>]
[GROUP BY <список-стовпчиків>]
[HAVING <умова_пошуку_груп>]
[ORDER BY <список-стовпчиків-для-сортування>]

Секція <список-повертаємих-стовпчиків> | *

“*” означає, що повернути потрібно всі стовпчики із заданого списка таблиць секції **FROM;**

<список-повертаємих-стовпчиків> – перераховує через кому імена стовпчиків, що містяться в таблицях, які перераховані в секції **FROM.** Стовпчики в результуючій таблиці будуть розміщені в порядку перерахування іх в **<список-повертаємих-стовпчиків>**. У випадку, якщо стовпчику в результуючій таблиці потрібно присвоїти нове ім’я, то можна зазначати імена стовпчиків наступним чином: **<ім’я-стовпчика> AS <ім’я-стовпчика-в-результуючій-таблиці>.**

Крім того, в якості елементів цього списку можуть бути *обчислювані значення елементів деяких із стовпчиків*. Для отримання значення таких стовпчиків потрібно вказати *вираз* його обчислення із значень, що зберігаються в БД.

Секція FROM <список-таблиць>

Задає список таблиць, перерахованих через кому, які беруть участь в запиті.

У випадку, якщо в декількох таблицях секції **FROM** присутнє однакове ім'я (або за бажанням) можна зазначати імена стовпчиків наступним чином: **<ім'я-таблиці>.<ім'я стовпчика>**, тут **<ім'я-таблиці>** повинно належати списку секції **FROM**.

Секція WHERE <умова-пошуку>

Для визначення умови відбору рядків, слід використати секцію **WHERE**. Рядок відбирається, якщо результат предиката, заданого в **<умова-пошуку>** має результат **TRUE.**

Розглянемо основні умови пошуку:

<вираз_1> = <вираз_2>

<вираз_1> <> <вираз_2>

<вираз_1> < <вираз_2>

<вираз_1> <= <вираз_2>

<вираз_1> > <вираз_2>

<вираз_1> >= <вираз_2>

Перевірка на належність діапазону значень

<перевіряємий-вираз> [NOT] BETWEEN <нижнє-значення> AND <верхнє-значення>

Перевірка на належність множині

<перевіряємий-вираз> [NOT] IN (<список-констант-через-кому>)

Перевірка на відповідність шаблону

<ім’я-стовпчика> [NOT] LIKE <шаблон> [ESCAPE <символ-пропуску>]

Перевірка на рівність значенню NULL

<ім’я-стовпчика> IS [NOT] NULL

Для побудови складних умов пошуку використовуються дужки та **AND**, **OR** та **NOT**.

Секція GROUP BY <список-стовпчиків>

Надає команду групувати рядки, що мають спільне значення елементів зазначених стовпчиків таким чином, щоб функція агрегації могла бути застосована до кожної групи.

Секція HAVING <умова_пошуку_груп>

Для визначення умови відбору груп рядків, слід використати секцію **HAVING**. Відбирається група, яка задовольняє умові. Умова аналогічна умові в секції **WHERE.**

Секція **HAVING** майже завжди використовується разом з секцією **GROUP BY**.

Секція ORDER BY <список-стовпчиків-для-сортування>

Використання цієї секції дозволяє відсортувати результати запиту. Тут

<список-стовпчиків-для-сортування> з перерахування через кому <елементів-списку-стовпчиків-для-сортування>.

<елемент-списку-стовпчиків-для-сортування> ::=

<ім'я-стовпчика>

[<порядковий-номер стовпчика>] [ASC | DESC]

Тут **ASC** – сортування в лексико-графічному порядку за зростанням, а **DESC** – за спаданням.

Приклад 4.

Додамо в таблицю авторів послідовно наступні записи:

INSERT INTO [AUTHORS]
([AU_NAME], [AU_INFO]) VALUES (‘Пушкін О.С.', '')

INSERT INTO [AUTHORS]
([AU_NAME], [AU_INFO]) VALUES (‘Шевченко Т.Г.', '')

Тепер знайдемо авторів у яких немає книжок в нашій бібліотеці:

```
DELETE FROM [AUTHORS]
WHERE [AU_ID] NOT IN
(SELECT [AB_AU] FROM [AUTHORS_BOOKS])
```

■

Додамо книгу:

```
INSERT INTO [BOOKS]
        ([BK_NAME], [BK_INFO], [BK_DC])
    VALUES ('Теорія алгоритмів', '', 2)
```

Та автора цій книзі:

```
INSERT INTO [AUTHORS_BOOKS]
        ([AB_BK], [AB_AU])
VALUES (5,4)
```

Знайдемо всіх авторів та їх кількість книг в бібліотеці, і відсортуємо список за алфавітом:

```
 SELECT [AU_NAME], COUNT([AB_BK]) AS [COUNT_BOOKS]
FROM [AUTHORS], [AUTHORS_BOOKS]
WHERE
 ([AU_ID] = [AB_AU])
GROUP BY [AU_ID], [AU_NAME]
ORDER BY [AU_NAME]
```

Результат виконання запиту:

AU_NAME	COUNT_BOOKS
Зубенко В.В.	1
Павріщева К.М.	1
Нікітченко М.С.	1
Омельчук Л.Л.	1
Шкільняк С.С.	2

INSERT INTO [AUTHORS]
([AU_NAME], [AU_INFO]) VALUES (‘Пушкін О.С.', '')

INSERT INTO [AUTHORS]
([AU_NAME], [AU_INFO]) VALUES (‘Шевченко Т.Г.', '')

AU_ID	AU_NAME	AU_INFO
1	Зубенко В.В.	АВТОР КНИГИ Програмування
2	Омельчук Л.Л.	АВТОР КНИГИ Програмування
3	Нікітченко М.С.	
4	Шкільняк С.С.	
5	Лавріщева К.М.	
20	Пушкін О.С.	
21	Шевченко Т.Г.	

Тепер знайдемо авторів у яких немає книжок в нашій бібліотеці:

```
SELECT [AU_ID], [AU_NAME] FROM [AUTHORS]
WHERE [AU_ID] NOT IN
(
    SELECT [AB_AU] FROM [AUTHORS_BOOKS]
)
```

AU_ID	AU_NAME
20	Пушкін О.С.
21	Шевченко Т.Г.

ЗАПИТ НА ВИДАЛЕННЯ РЯДКІВ DELETE

<delete> ::= DELETE FROM <ім’я-таблиці>

[WHERE <умова-пошуку>]

Секція [WHERE <умова-пошуку>] аналогічна до такої секції в запиті на пошук.

Для його видалення достатньо виконати один з наступних запитів:

```
DELETE FROM [AUTHORS] WHERE (([AU_ID] = 20)
OR([AU_ID] = 21))
```

Або

```
DELETE FROM [AUTHORS] WHERE (([AU_NAME] = 'Пушкін
О.С.‘) OR ([AU_NAME] = ‘Шевченко Т.Г.’))
```

Нехай потрібно видалити авторів, книжок яких в бібліотеці немає:

```
DELETE FROM [AUTHORS]
WHERE [AU_ID] NOT IN
(SELECT [AB_AU] FROM [AUTHORS_BOOKS])
```

ЗАПИТ НА ОНОВЛЕННЯ ІСНУЮЧИХ ДАНИХ UPDATE

<update> ::= UPDATE <ім'я-таблиці>

SET <список-пар-ім'я-поля-та-вираз>

[WHERE <умова-пошуку>]

Секція [WHERE <умова-пошуку>] аналогічна до такої секції в запиті на пошук.

Помістимо в поле **DA_INFO** таблиці **DIC_AUTHOR** для авторів книги 1 відповідну інформацію:

Приклад ?.2.

```
UPDATE [AUTHORS]
SET [AU_INFO] = 'АВТОР КНИГИ Програмування'
WHERE [AU_ID] IN
  (
    SELECT [AB_AU] FROM [AUTHORS_BOOKS] WHERE [AB_BK]=1
  )
```

Після виконання цього запиту маємо наступні дані в таблиці **AUTHORS:**

AU_ID	AU_NAME	AU_INFO
1	Зубенко В.В.	АВТОР КНИГИ Програмування
2	Омельчук Л.Л.	АВТОР КНИГИ Програмування
3	Нікітченко М.С.	
4	Шкільняк С.С.	
5	Лавріщева К.М.	